(19) Europäisch s Patentamt
Europ an Patent Offic
Offi e européen des br v ts

(11) EP 0 747 831 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.12.1996 Bulletin 1996/50

(51) Int Cl.[6]: **G06F 13/40**

(21) Application number: **96480081.7**

(22) Date of filing: **31.05.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **07.06.1995 US 472603**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

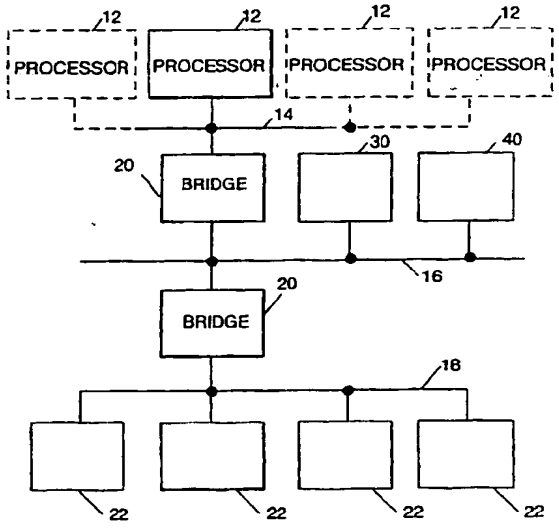
(72) Inventors:
- **Neal, Dan M.**
  Round Rock, Texas (US)
- **Silha, Edward J.**
  Austin, Texas 78759 (US)
- **Thurber, Steven M.**
  Austin, Texas (US)

(74) Representative: **Schuffenecker, Thierry**
Compagnie IBM France,
Département de Propriété Intellectuelle
06610 La Gaude (FR)

(54) Data processing system including buffering mechanism for inbound and outbound reads and posted writes

(57) A data processing system includes a host processor, a number of peripheral devices, and one or more bridges which may connect between the host, peripheral devices and other hosts or peripheral devices such as in a network. Each bus to bus bridge connects between a primary bus and a secondary bus wherein for the purpose of clarity, the primary bus will be considered as the source for outbound transactions and the destination for inbound transactions and the secondary bus would be considered the destination for outbound transactions and the source for inbound transactions. Each bus to bus bridge includes an outbound data path, an inbound data path, and a control mechanism. The outbound data path includes a queued buffer for storing transactions in order of receipt from the primary bus where the requests in the queued buffer may be mixed as between read requests and write transactions, the outbound path also includes a number of parallel buffers for storing read reply data and address information. The inbound path is a mirror image of the outbound path with read requests and write requests being stored in a sequential buffer and read replies being stored in a number of parallel buffers. Both the inbound path and the outbound path in the bus to bus bridge are controlled by a state machine which takes into consideration activity in both directions and permits or inhibits bypass transactions.

FIG. 1



EP 0 747 831 A2

Description

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to data processing systems, and more particularly to data processing systems including mechanisms for efficiently handling inbound and outbound reads and posted writes from host to peripheral devices.

1. PRIOR ART

Many standard bus architectures for use in data processing systems such as the Peripheral Component Interconnect (PCI) bus specification contain ordering rules for transactions that traverse a bus to bus bridge. Examples of such transactions are read requests outbound from a host processor to a peripheral device, read replies containing the data from the peripheral device inbound to the host processor or posted writes in which a write command is posted to a buffer in a bus to bus bridge allowing the host to proceed with other processing. A posted write herein means that the write has completed on the initiating bus and is posted in the bus to bus bridge for future completion on the destination bus. Current bus to bus bridge architecture requires write buffers in both outbound (away from the host processor) and inbound (towards the host processor) direction to be flushed out prior to completing a read transaction that traverses the bus to bus bridge. The requirement to flush the buffers can create a performance problem for a busy server by holding up the processor read accesses which are Retried if there are any posted writes in the bridge buffers. Retry herein means that the target device being addressed by the master acknowledges the access but signals that it is busy and it terminates the transaction. The master then tries the access again at a later time.

Several prior art bridge architectures attempt to improve performance and eliminate throughput delays.

An article in the IBM Technical Disclosure Bulletin dated July, 1992, at page 233 and following entitled "Architecture for High Performance Transparent Bridges," shows architecture for high performance bridges with multi-port bridges interconnecting networks including buffering components which guarantee transfer of data whether read or write with uniform handling inside the bridge.

Although the article generally relates to improvement in performance of bridges, the article does not address the problem of requiring buffers to be flushed out prior to completing a read transaction which degrades system performance.

U. S. Patent 5,XXX,XXX(BC993012) entitled "Method and Apparatus for Configuring PCI Bridges on Multiple PCI Buses Environment," shows a method and apparatus for configuring PCI bridges. The patent also shows direct connection to the host system through host bridges with bus to bus bridges and peripheral buses directly connected to a system bus.

While the patent teaches the operation of the industry standard PCI bridge architecture, it does not address nor suggest any solution to the performance problem created by the requirement to flush buffers prior to a read transaction.

U. S. Patent 5,333,269 entitled "Mechanism for Transferring Messages Between Source and Destination Users Through a Shared Memory," teaches a common bus to which a memory with a number of independent buffers, a memory interface and a central control unit are connected. The memory interface receives messages from source users, stores the message in selected buffers, and chains the buffers together. The control apparatus generates inbound message cues and outbound message cues in response to commands which it receives from the memory interface.

Although the patent has some similarities to high performance bus bridge architectures, the patent does not address nor suggest a solution to the problem of requirement for flushing out buffers prior to completing a read transaction which follows a posted write transaction.

U. S. Patent 5,247,620 entitled "Bridge Apparatus with an Address Check Circuit for Interconnecting Networks," teaches multiple inbound and outbound buffers for reading out of information from processor to peripheral devices and for bridging in a network environment.

Although the patent shows in Figure 1 a bus to bus bridge including a buffer memory, the patent does not teach nor suggest a mechanism for improving performance in bus to bus bridges which eliminates the need for flushing out buffers between write and read transactions.

Other prior art patents and publications show various bridges and buffering mechanisms, but none of the prior art teaches nor suggests the performance improvement of eliminating the requirement for flushing out buffers for a read operation following a write operation.

Further, the prior art does not adequately deal with the problem of handling multiple mixed inbound and outbound transactions.

SUMMARY OF THE INVENTION

Therefore, it is an object of the present invention to efficiently handle multiple mixed transactions in both inbound and outbound directions in a bus to bus bridge having a control mechanism for controlling the transmission of read requests, read replies and posted writes in an efficient manner while maintaining a high performance level of the system.

Accordingly, a data processing system includes a host processor, a number of peripheral devices, and one or more bridges which may connect between the host,

peripheral devices and other hosts or peripheral devices such as in a network. Each bus to bus bridge connects between a primary bus and a secondary bus wherein for the purpose of clarity, the primary bus will be considered as the source for outbound transactions and the destination for inbound transactions and the secondary bus would be considered the destination for outbound transactions and the source for inbound transactions. Different transactions on the primary "initiating" bus may or may not occur simultaneous to other transactions on the "secondary" bus, depending on implementation. Each bus to bus bridge includes an outbound data path, an inbound data path, and a control mechanism. The outbound data path includes a queued buffer for storing transactions in order of receipt from the primary bus where the requests in the queued buffer may be mixed as between read requests and write transactions, the outbound path also includes a number of parallel buffers for storing read reply data and address information.

The inbound path is a mirror image of the outbound path with read requests and write requests being stored in a sequential buffer and read replies being stored in a number of parallel buffers. Both the inbound path and the outbound path in the bus to bus bridge are controlled by a state machine which takes into consideration activity in both directions and permits or inhibits bypass transactions. Bypass herein means that a subsequent transaction may be allowed to be processed before a prior transaction, even though the prior transaction entered the queue earlier than the subsequent transaction.

It is a feature of the present invention that inbound and outbound transaction requests may be handled in nonsequential order under the control of a predetermined state machine.

It is another feature of the present invention that inbound and outbound transactions in a bus to bus bridge can be efficiently handled by a mechanism which allows read transactions to follow write transactions without requiring buffers to be flushed out.

It is another feature of the present invention that proper data consistency be maintained relative to the different transactions as they traverse a bus to bus bridge.

Other transactions such as Read requests must be completed (data fetched) on the destination bus prior to completing on the initiating bus. With the features of the present invention, processing of transactions such as a Read access can progress without flushing Posted Writes.

A Posted Read herein means that the target acknowledges the access and signals that it is busy and then terminates the access with Retry. The master will try the access again later. The bus to bus bridge then posts the Read request for subsequent processing such that it may have the Read data available when the master tries the access again later. If the master tries again before the bus to bus bridge obtains the requested data, the bridge again signals Retry but does not buffer a second copy of the access request.

These techniques can be applied to host bus bridges and to adapter devices that also post write and read transactions in addition to bus to bus bridges.

The foregoing has broadly outlined the features and technical advantages of the present invention in order that the detailed description of the invention which follows may be better understood. Additional features and advantages of the invention will be described hereinafter which form the subject of the claims of the invention.

BRIEF DESCRIPTION OF THE DRAWING

For a more complete understanding of the present invention, and the advantages thereof, reference will be made to the following description taken in conjunction with the accompanying drawing, in which:

Figure 1 is a block diagram of a data processing system embodying the present invention.

Figure 2 is a block diagram of a bus to bus bridge illustrating features of this invention in accordance with the system of Figure 1.

Figure 3 is a logic table showing control of inbound and outbound transactions in the bridge of Figure 2 for each type of transaction on both the outbound and the inbound path.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

Referring now to Figure 1, a data processing system embodying the present invention will be described. System 10 includes a host processor 12 connected to a first bus 14. Other processors 12 shown in dotted lines on Figure 1 may also be connected to bus 14.

Also connected to bus 14 is bus to bus bridge 20 which connects between first bus 14 and a second bus 16. Other devices 30, 40 may be connected to bus 16 as well. Additionally, another bridge 20 is connected to bus 16 and to a third bus 18. Also connected to bus 18 may be a number of stations or peripheral devices 22. Other than bridge 20, each of the elements referenced above with respect to Figure 1 are all well known in the art and need not be described in greater detail herein. As an example, but in no way limiting the scope of the invention, buses 14, 16, and 18 each may be a PCI bus which has become an industry standard.

Referring now to Figure 2, bus to bus bridge 20 will be described in greater detail. As used in this description of the preferred embodiment of the invention, the term outbound refers to those transactions which are away from the processor and inbound refers to those transactions which are toward the processor.

Bridge 20 has two data paths, outbound path 202 and inbound path 204. All transactions inbound and outbound are controlled by state machine 206.

When an outbound transaction is received by transaction router 208, the transaction format is examined to

determine whether the transaction is a read request, a read reply, or a write request.

A read request has a format of an address and control information. A read reply has a format which includes an address, control information and the data which had been requested and which is now being sent to the original requestor.

A write transaction includes an address, control information and the data which will be written into the selected address.

Assuming, for example, that the transaction currently active in transaction router 208 is a read request, the transaction will be forwarded to one of a number of sequential buffers 210, 212, 214. The sequential buffer entries 210, 212, 214 may be any appropriate number of buffer entries in length adequate to handle the stack of transactions anticipated to be pending in bridge 20.

If the transaction pending in transaction router 208 is a read reply, the read reply transaction is transmitted to one of parallel buffer entries 216, 218.

At any time, the transaction to be selected and passed to a secondary bus by transaction selection circuit 220 is controlled by state machine 206. This allows, for example, out of sequence accesses to transactions stored in sequential buffers 210, 212 or 214 or to a read reply transaction in parallel buffers 216, 218.

The inbound path 204 includes the same elements as outbound path 202 except the elements are connected in the reverse order. That is, transaction router 222 is connected to the secondary bus 16 or 18 (see FIGURE 1). As explained above with reference to outbound transaction router 208, inbound transaction router 222 is controlled by state machine 206. Each transaction is routed to one of the sequential inbound buffers 224, 226, and 228 for read request and write transactions. Read reply transactions are transmitted to parallel buffers 232, 234. Inbound selection circuit 230 under the control of state machine 206 controls transmission of transactions from either sequential buffers 224, 226 or 228 or from parallel buffers 232 or 234 respectively.

State machine 206 operates under a protocol which is defined by the state machine logic table shown in Figure 3 to control multiple transactions flowing either outbound or inbound or both.

Referring now to Figure 3, the logic table which controls state machine 206 will be described. Across the top of the logic table are transactions which represent either outbound or inbound transactions and which will be referred to for convenience as a first transaction. Along the left side of the logic table are outbound and inbound transactions which will be referred to for convenience as second transactions.

If, for example, there is an outbound first transaction which is a read request, and an outbound second transaction which is a read request, the intersection of the column and row in the logic table indicate a one. A one in any position in the logic table indicates a permitted out of sequence bypass transaction. A zero in any position of the logic table of Figure 3 indicates a prohibited bypass transaction. Thus, an outbound read request as a first transaction followed by an outbound read request as a second transaction are permitted to be bypassed by the second transaction. To bypass means to complete on the destination bus a second request prior to completion of a first request. However, if the first transaction is, for example, an outbound write transaction, bypass of an outbound read request is prohibited.

It should be noted that all positions in the logic table where there is a zero indicating a prohibited bypass transaction involve write transactions. For example, an outbound Read request cannot bypass an outbound write, nor can an outbound Read reply or write bypass another outbound write. For inbound transactions, the same holds true. Thus, an inbound Read request, Read reply or write transaction may not bypass an earlier inbound write transaction. This logic table which controls state machine 206 controls the gating of transactions in transaction routers 208 and 222 and transaction selection circuits 220 and 230 respectively.

An outbound read request may bypass earlier outbound read requests. Outbound read requests may bypass earlier outbound read replies. Outbound read requests may not bypass earlier write transactions. This is to prevent a read of a register previously written, from being performed before the write transaction arrives thus causing the reading of incorrect data.

Outbound read replies may bypass earlier outbound read requests or read reply transactions. For a generalized bus to bus bridge, an outbound read reply may not bypass an earlier outbound write transaction to prevent an interrupt arriving early indicating a device has completed a memory write allowing access to the subject write data, before the write to memory is performed. In a less general system where this interrupt race is only of concern for the inbound case, then an outbound read reply may be allowed to bypass an earlier write transaction.

Outbound write transactions may bypass earlier outbound read requests. This is due to the fact that a Read request is only a request until it completes and has not occurred in terms of data ordering, therefore, a Write transaction may be allowed to bypass the Read request.

Outbound write transactions may not bypass earlier outbound write transactions. This is to assure that write transactions on one bus that also traverse a bridge complete in the same order on both buses.

Relative to inbound versus inbound transactions, an inbound read request may bypass earlier inbound read requests. Inbound read requests may bypass earlier inbound read replies.

Inbound read requests may not bypass earlier inbound write transactions to prevent a read of a register previously written from being performed before the write data arrives.

Inbound read replies may bypass earlier inbound

read requests and read replies. Inbound read replies may not bypass earlier inbound write transactions to prevent an interrupt arriving early indicating a device has completed a write to system memory and the processor reading the memory location before the subject write to memory is performed.

Inbound Write transactions may bypass inbound Read replies and Read requests.

Inbound write transactions may not bypass earlier inbound write transactions. As before, this is to assure that write transactions on one bus that also traverse a bridge complete in the same order on both buses.

It should be noted that any outbound transaction may bypass an earlier inbound transaction and that any inbound transaction may bypass any earlier outbound transaction. This is possible since these transactions are directed to different address spaces. For cases where any outbound versus inbound transaction or any inbound versus outbound transaction requires ordering, it is handled by software protocols beyond the scope of this invention.

A master that attempts a read access and is retried, must periodically try the access again until it completes.

If non-prefetchable read reply data remains in a buffer in the bus to bus bridge beyond its timeout period, the bus to bus bridge activates a system error, signalling to the system that the bridge has a stalled buffer of non-prefetchable read data.

Although the present invention and its advantages have been described in detail, it should be understood that various changes, substitutions and alterations can be made herein without departing from the spirit or scope of the invention as defined in the appended claims.

Claims

1. A data processing system, comprising:

   One or more processors;

   One or more peripheral devices;

   A plurality of buses connecting said one or more processors and said one or more peripheral devices through one or more bridges; and

   One or more bridges for controlling transactions between a first bus and a second bus of said plurality of buses, each said bridge comprising:

   A first data path for handling transactions from said first bus to said second bus;

   A second data path for handling transactions from said second bus to said first bus; and

   A controller for controlling gating and sequence of transactions between said first bus and said second bus through said first data path and said second data path.

2. A data processing system according to claim 1, wherein said first data path and said second data path each further comprise:

   A transaction router circuit connected to an input to said bridge for gating transactions to buffers in said data path;

   One or more buffers connected to said transaction router circuit for storing read reply transactions;

   One or more buffers connected to said transaction router circuit for storing read request transactions and write transactions;

   A transaction selection circuit connected to outputs of said buffers for selecting one or more transactions to be output from said data path under the control of said controller.

3. A data processing system according to claim 1 wherein said controller further comprises a state machine for controlling transactions through said first data path and said second data path.

4. A data processing system according to claim 3, wherein said state machine controls gating of transactions in said first and second data path in accordance with a predetermined logic structure.

5. A data processing system, according to claim 4, wherein said predetermined logic structure further comprises:
   means for controlling said read transactions and said write transactions to eliminate flushing write buffers prior to read transactions being gated through said controller.

6. A data processing system according to claim 1 wherein said first data path is an outbound data path.

7. A data processing system according to claim 1 wherein said second data path is an inbound data path.

8. A bus to bus bridge, comprising:
   First and second data paths each operating under the control of a controller for controlling gating and sequence of transactions between a first bus and a second bus through said first data path and said second data path.

9. A bus to bus bridge according to claim 8, wherein said controller further comprises a state machine operating in accordance with a predetermined logic structure; and

wherein each said data path further comprises:

a transaction routing circuit connected to an input bus for gating transactions to one of a plurality of buffers in accordance with control signals from said controller;

a plurality of buffers for storing said transactions, a first plurality of said buffers storing read reply transactions, and a second plurality of said buffers storing read request transactions and write transactions; and

a transaction selection circuit connected between outputs of said buffers and an output bus for gating transactions to said output bus under the control of the logic structure of said controller.

10. A method, in a data processing system, for controlling a plurality of transactions from a first bus to a second bus through a bridge, comprising the steps of:

Determining a type of transaction from said first bus in a transaction routing circuit;

Gating a transaction to one of a plurality of buffers depending on said transaction type;

Controlling outputs of said buffers in accordance with a predetermined logic structure; and

Gating a selected transaction to said second bus; and

further comprising the step of gating a second received transaction ahead of a first received transaction under control of said logic structure.

FIG. 1

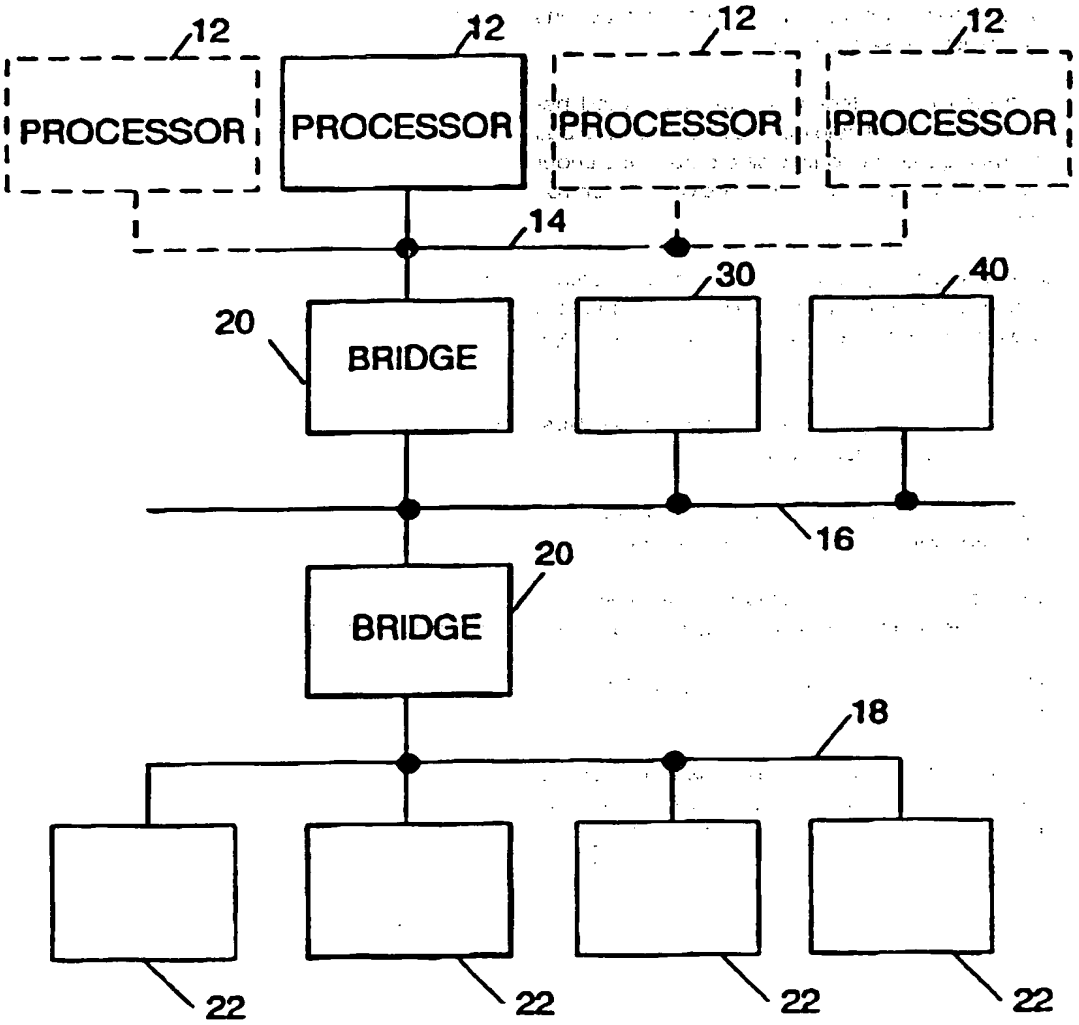
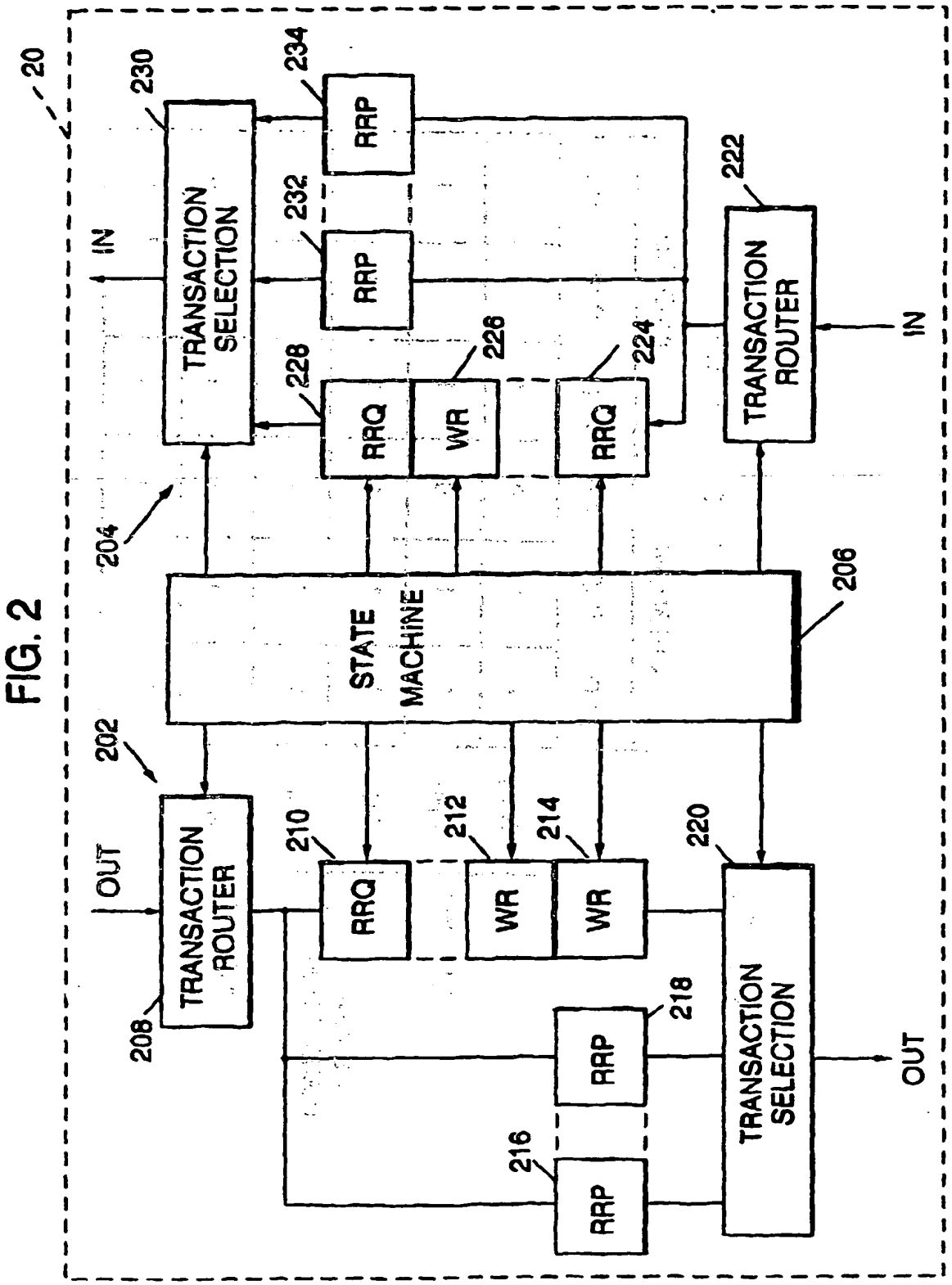


FIG. 2



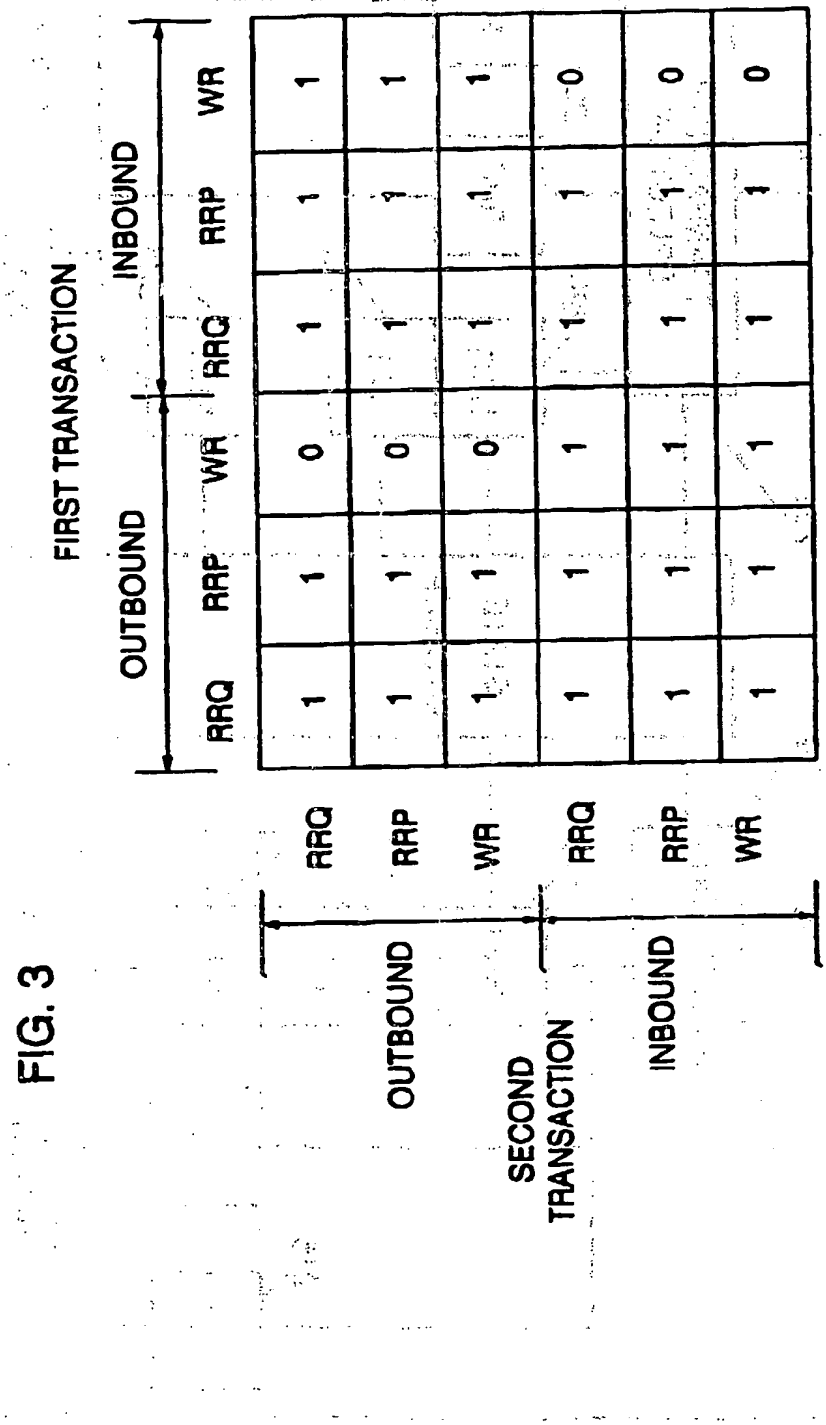


FIG. 3